《第七八章 流密码》示例代码

作者：韩露露、杨波

日期：2019年3月1日

说明

本电子文档来源于书籍《深入浅出CryptoPP密码学库》，它最初被存放于GitHub上。任何人都可以复制、传播、使用本示例代码。

⇓

简介

《深入浅出CryptoPP密码学库》内容简介：

本书向读者介绍密码学库CryptoPP（或Crypto++）的使用方法和设计原理。CryptoPP是一个用C++语言编写的、开源的、免费的密码程序库，它最初由Wei Dai开发，现由开源社区维护。CryptoPP库广泛应用于学术界、开源项目、非商业项目以及商业项目，它几乎包括了目前已经公开的所有密码算法，支持当前主流的多种系统平台，并且具有良好的设计结构和较高的执行效率。

全书共15章，主要内容包括随机数发生器、Hash函数、流密码、分组密码、消息认证码、密钥派生和基于口令的密码、公钥加密系统、数字签名、密钥协商等，本书涵盖C++程序设计、设计模式、数论和密码学等知识。

本书最大的特点就是以应用为导向、以解决实际工程问题为目标，理论结合实践，将抽象的密码学变成保障信息安全的实际工具。

本书可以作为密码学、网络安全等专业在校学生的上机实验教材，也可以作为信息安全产品开发者、科研人员、密码算法实现者的参考手册。

⇓

资源

本书更多示例代码：https://github.com/locomotive-crypto
Crypto++网站：https://www.cryptopp.com/
Crypto++库GitHub地址：https://github.com/weidai11/cryptopp
Crypto++库SourceForge地址：https://sourceforge.net/projects/cryptopp/
Crypto++库Google论坛：
⇒公告通知地址：https://groups.google.com/forum/#!forum/cryptopp-announce
⇒用户群组地址：https://groups.google.com/forum/#!forum/cryptopp-users

目录

1 使用XSalsa20算法加解密字符串

下面以流密码XSalsa20算法为例，演示加密和解密字符串的一般方法。

```cpp
#include<iostream> //使用cin、cout
#include<salsa.h> //使用XSalsa20算法
#include<osrng.h> //使用AutoSeededRandomPool算法
#include<secblock.h> //使用SecByteBlock
using namespace std; //std是C++的命名空间
using namespace CryptoPP; //CryptoPP是CryptoPP库的命名空间
//功能：以十六进制的形式将pb缓冲区的数据打印至标准输出设备
//参数pb：待打印输出的缓冲区首地址
//参数length：缓冲区pb的长度（字节）
//返回值：无
void printHex(byte* pb,size_t length);
int main()
{
    XSalsa20::Encryption enc; //定义加密器对象
    XSalsa20::Decryption dec; //定义解密器对象
    cout << "缺省密钥长度（字节）:"
        << enc.DefaultKeyLength() << endl;
    cout << "最小的密钥长度（字节）:"
        << enc.MinKeyLength() << endl;
    cout << "最大的密钥长度（字节）:"
        << enc.MaxKeyLength() << endl;
    cout << "缺省的初始向量大小（字节）:"
        << enc.DefaultIVLength() << endl;
    cout << "最小的初始向量大小（字节）:"
        << enc.MinIVLength() << endl;
    cout << "最大的初始向量大小（字节）:"
        << enc.MaxIVLength() << endl << endl;
    //定义一个随机数发生器，用于产生随机的密钥Key和初始向量IV
    AutoSeededRandomPool prng; //定义随机数发生器对象
    //plain：存储待加密的明文，
    //cipher：存储加密后的密文，
    //recover：存储解密后的明文
    string plain("I like cryptography very much"),cipher,recover;
    //动态申请空间以存储接下来生成的密钥key和初始向量iv
    SecByteBlock key(enc.DefaultKeyLength()),iv(enc.DefaultIVLength
        ());
    prng.GenerateBlock(key,key.size()); //产生随机的密钥key
    prng.GenerateBlock(iv,iv.size()); //产生随机的初始向量iv
    cout << "key:";
    printHex(key,key.size()); //以十六进制的形式打印key
    cout <<endl<< "IV:";
    printHex(iv,iv.size()); //以十六进制形的式打印IV
    //加密和解密准备
```

```
43      //设置加密器的密钥key和初始向量iv
44      enc.SetKeyWithIV(key,key.size(),iv,iv.size());
45      //设置解密器的密钥key和初始向量iv
46      dec.SetKeyWithIV(key,key.size(),iv,iv.size());
47      cipher.resize(plain.size()); //分配存储空间
48      //执行加密
49      enc.ProcessData((CryptoPP::byte*)&cipher[0],(CryptoPP::byte*)
            plain.c_str(),plain.size());
50      cout << endl << "plain:" ;
51      //以十六进制的形式打印明文
52      printHex((CryptoPP::byte*)plain.c_str(),plain.length());
53      cout << endl << "cipher:";
54      //以十六进制的形式打印密文
55      printHex((CryptoPP::byte*)&cipher[0],cipher.size());
56      //执行解密操作
57      recover.resize(cipher.size()); //分配存储空间
58      dec.ProcessData((CryptoPP::byte*)&recover[0],(CryptoPP::byte*)
            cipher.c_str(),cipher.size());
59      cout << endl << "recover:";
60      //以十六进制的形式输出解密后的明文
61      printHex((CryptoPP::byte*)&recover[0],recover.size());
62      cout << endl;
63      return 0;
64  }
65  void printHex(byte* pb,size_t length)
66  {//以十六进制输出pb缓冲区的数据
67      for(size_t i=0; i < length; ++i)
68      {
69          printf("%02X",pb[i]); //也可以使用C++的操作方式
70      }
71  }
```

执行程序，程序的输出结果如下：

```
缺省密钥长度（字节）:32
最小的密钥长度（字节）:32
最大的密钥长度（字节）:32
缺省的初始向量大小（字节）:24
最小的初始向量大小（字节）:24
最大的初始向量大小（字节）:24
key:33A086EFE0E2491FAC824556C4EDFE3903CADD16F0FEF6A25A589B43BF92B976
IV:4F5D2208EFF0E9C6E5D36352CF33FC0CDF8FAD3257C04D3D
plain:49206C696B652063727970746F6772617068792076657279206D756368
cipher:2D99EDD93F0E2BF2C14BD43E4F328F033AD99469B683FD7E96AB7F6337
recover:49206C696B652063727970746F6772617068792076657279206D756368
请按任意键继续. . .
```

2 使用ChaCha20算法加解密文件

下面以流密码算法ChaCha20为例，演示加密和解密文件的方法。

```
#define _CRT_SECURE_NO_WARNINGS
#include<iostream> //使用cin、cout
#include<chacha.h> //使用ChaCha20算法
#include<osrng.h> //使用AutoSeededRandomPool算法
#include<secblock.h> //使用SecByteBlock
#include<string> //使用string
using namespace std; //std是C++的命名空间
using namespace CryptoPP; //CryptoPP是CryptoPP库的命名空间
//功能：使用流密码算法将文件srcfile中的内容加密或解密，
//  并把加密或解密结果存储于文件desfile
//参数srcfile：待加密或解密的文件
//参数encdec：使用的流密码算法
//参数desfile：存放加密或解密结果的文件
//返回值：无
void EncOrDecFile(const string& srcfile,StreamTransformation& encdec
    ,const string& desfile);
int main()
{
    ChaCha20::Encryption enc; //定义加密器对象
    ChaCha20::Decryption dec; //定义解密器对象
    //定义一个随机数发生器，用于产生随机的密钥Key和初始向量IV
    AutoSeededRandomPool prng; //定义随机数发生器对象
    //动态申请空间以存储接下来生成的密钥key和初始向量iv
    SecByteBlock key(enc.DefaultKeyLength()),iv(enc.DefaultIVLength
        ());
    //产生随机的key和iv
    //利用随机数发生器产生随机的密钥key
    prng.GenerateBlock(key,key.size());
    //利用随机数发生器产生随机的初始向量iv
    prng.GenerateBlock(iv,iv.size());
    //加密和解密准备
    //设置加密器的密钥key和初始向量iv
    enc.SetKeyWithIV(key,key.size(),iv,iv.size());
    //设置解密器的密钥key和初始向量iv
    dec.SetKeyWithIV(key,key.size(),iv,iv.size());
    //执行加密：将文件"stream_02.cpp"中的数据加密后存放于文件"cipher.txt"中
    EncOrDecFile("stream_02.cpp",enc,"cipher.txt");
    //执行解密：将文件"cipher.txt"中的数据加密后存放于文件"recover.txt中
    EncOrDecFile("cipher.txt",dec,"recover.txt");
    return 0;
}
void EncOrDecFile(const string& srcfile,StreamTransformation& encdec
    ,const string& desfile)
```

```
41 {//用流密码对象encdec实现对文件srcfile的加密或解密操作,
42 //   并将操作的结果存放于文件desfile
43     //打开待加密或解密的文件,也可用C++的文件操作方式
44     FILE* infp = fopen(srcfile.c_str(),"rb");
45     //打开存放操作结果的文件
46     FILE* outfp = fopen(desfile.c_str(),"wb");
47     if(!infp || !outfp) //文件打开错误
48     {
49         cout << "文件打开失败" << endl;
50         return ;
51   }
52     int readlength;
53     SecByteBlock readstr(1024); //动态申请1024个字节的存储空间
54     while(!feof(infp))
55     {//文件没有读取结束
56         //从待加密或解密的srcfile文件中读取数据
57         readlength=fread(readstr,sizeof(SecByteBlock::value_type),
               readstr.size(),infp);
58         //确定实际读取到的数据长度
59         readlength = readlength < readstr.size() ? readlength:
               readstr.size();
60         encdec.ProcessString(readstr,readlength); //执行加密或解密操作
61         //将加密或解密的结果存放于desfile文件中
62         fwrite(readstr,sizeof(SecByteBlock::value_type),readlength,
               outfp);
63     }
64     fclose(infp); //关闭文件
65     fclose(outfp); //关闭文件
66 }
```

执行该程序，待程序运行完毕后。当前工程的目录下面会产生两个文件，它们的名字分别是cipher.txt和recover.txt，如图1所示。其中，cipher.txt是stream_02.cpp文件被加密后对应的密文文件。recover.txt是cipher.txt文件被解密后对应的明文文件。

名称	修改日期	类型	大小
Release	2018/9/30 21:23	文件夹	
cipher.txt	2018/12/6 11:14	文本文档	3 KB
CryptoPPRelease.props	2018/8/8 17:41	PROPS 文件	1 KB
recover.txt	2018/12/6 11:14	文本文档	3 KB
stream_02.cpp	2018/9/14 19:10	C++ Source	3 KB
stream_02.vcxproj	2018/9/14 16:36	VC++ Project	5 KB
stream_02.vcxproj.filters	2018/9/14 16:36	VC++ Project Fil...	1 KB

图1 当前工程所在目录下的内容

3 以Pipeling范式方式使用ChaCha12算法

下面以流密码算法ChaCha12为例，演示如何用CryptoPP库的Pipeling范式技术实现文件的加解密。

```
#include<iostream> //使用cin、cout
#include<chacha.h> //使用ChaCha12算法
#include<osrng.h> //使用AutoSeededRandomPool算法
#include<secblock.h> //使用SecByteBlock
#include<string> //使用string
#include<files.h> //使用FileSource、FileSink
#include<filters.h> //使用StreamTransformationFilter
using namespace std; //std是C++的命名空间
using namespace CryptoPP; //CryptoPP是CryptoPP库的命名空间
int main()
{
    ChaCha12::Encryption enc; //定义加密器对象
    ChaCha12::Decryption dec; //定义解密器对象
    //定义一个随机数发生器，用于产生随机的密钥Key和初始向量IV
    AutoSeededRandomPool prng; //定义一个随机数发生器对象
    //动态申请空间以存储接下来生成的密钥key和初始向量iv
    SecByteBlock key(enc.DefaultKeyLength()),iv(enc.DefaultIVLength
        ());
    //产生随机的key和iv
    prng.GenerateBlock(key,key.size()); //产生随机的加解密密钥key
    prng.GenerateBlock(iv,iv.size()); //产生随机的初始向量IV
    //加密和解密准备
    //设置加密器对象的加密密钥key和初始向量iv
    enc.SetKeyWithIV(key,key.size(),iv,iv.size());
    //设置解密器对象的解密密钥key和初始向量iv
    dec.SetKeyWithIV(key,key.size(),iv,iv.size());
    //加密文件-将文件stream_03.cpp用enc流密码加密，并把密文存储于cipher.txt文件中
    FileSource fenc("stream_03.cpp",true,
        new StreamTransformationFilter(enc,
            new FileSink("cipher.txt")));
    //解密文件-将文件cipher.txt用dec流密码解密，并把明文存储于recover.txt文件中
    FileSource fdec("cipher.txt",true,
        new StreamTransformationFilter(dec,
            new FileSink("recover.txt")));
    return 0;
}
```

本示例与示例二的目标是一样的，它们都实现了对文件的加解密。所不同的是，第一，示例三使用了Pipeling范式技术。第二，当以这种方式使用流密码算法执行加密时，我们还可以选择预定义的填充方案。在本示例程序中，我们采用了缺省的填充方案。

4 声明

Cryptography

⇓

⇓

⇓

此为《深入浅出CryptoPP密码学库》随书电子文档，它仅包含书籍中示例程序的源代码。关于示例代码的解释说明，详见书籍相应章节内容。

由于作者水平有限，错误之处在所难免。欢迎通过如下方式反馈相关问题：

⇒ QQ：1220195669

⇒ 微信：cc1220195669

⇓

⇓

⇓

《深入浅出CryptoPP密码学库》